Supplementary proceedings

Where are Mifrenz?

Dr Tim D. Hunt
Waikato Institute of Technology (Wintec)
Tristram Street
Private Bag 3036
Hamilton 2020
New Zealand
tim.hunt@wintec.ac.nz

Abstract

This paper describes the further development of an email application for children, called Mifrenz, in particular the implementation of the ability to use the application from more than one physical location. During evaluation of the previous version of the software, it was discovered that many children wish to access their email from multiple physical locations. From this problem the following objectives for this work were identified: 1) Provide access to sent and received email from multiple locations, 2) Provide access to persistent data (contacts and jokes) and 3) Keep the development time to a minimum. During the design stage the use of the IMAP technology was identified as the only practical solution for sharing messages. The design stage also identified IMAP and Google Docs as two methods for solving the storage of persistent data problem. Although both of these designs were shown to work during the development stage, IMAP was eventually selected as the most appropriate technology for storing persistent data. The Design Science Research Process (Peffers, 2006) is used as a framework in describing this work. The steps of this process framework are: 1) Problem identification and motivation, 2) Objectives of a solution, 3) Design and development, 4) Demonstration, 5) Evaluation and 6) Communication.

Keywords: Mifrenz, Email, Children, IMAP.

1 Introduction

Email has become a common means of communication among adults. The rise in popularity of email has also been accompanied by a rise in the amount of unsolicited email, known as SPAM. Although SPAM is normally a mere nuisance to adults, it is a major concern for parents when their children wish to use email unsupervised. To alleviate the problem of SPAM, Mifrenz 0.1 (Hunt, 2007) was designed and developed to allow young children to safely use email with a minimum of adult supervision.

This Supplementary Proceedings paper appeared at the *21st Annual Conference of the National Advisory Committee on Computing Qualifications* (NACCQ 2008), Auckland, New Zealand. Samuel Mann and Mike Lopez (Eds).

During end user evaluation of Mifrenz 0.1, it became clear that the application was not suitable for children who wished to access their email from two separate addresses. This paper describes this problem and the investigation and implementation of a suitable solution.

The paper format follows the Design Science Research Process framework as proposed by Peffers (2006). This framework suggests the following 6 steps:

- Problem identification and motivation
- Objectives of a solution
- Design and development
- Demonstration
- Evaluation
- Communication

Although it is suggested that progress is made from one step to the next, you do not necessary have to start at the first step, but could enter the framework at any of the first 4 steps. Also the framework is an iterative process in that you may return to an early step as the particular situation deems necessary.

This paper describes the implementation of a solution, based on the IMAP (The IMAP Connection, 2008) protocol that allows sharing of information across multiple physical locations. It may be of interest to those who teach computer program design and would like a case study on how a design issue was presented and eventually solved.

The rest of this introduction gives an overview of current email options for children followed by a brief overview of Mifrenz 0.1 to place the current work in context.

1.1 Current email options for children

A number of alternative solutions exist for children to use email and are presented here.

1.1.1 Standard adult software with parent supervision

The first solution is to allow children to use a standard email application. A number of scenarios exist, none of which are deemed very suitable.

- The parent allows the child to use their own email account. They may 'hover' close by as the child writes or reads an email.
- A child has their own email address but the parent closely controls access to when the child reads the email (by not giving the child the password).

- The parent sets up a 'white list' and sets an option to have email from non contacts to be deleted. The problem with this solution is that the child may choose to stop using the white list.

1.1.2 Specific server based email server for children

Kidmail.net (2007) is an example of a solution that relies on a dedicated server to provide a web based email application. This allows a high quality solution as control of various options can be set at an appropriate level for the age of the child. Child access to changing these options is prevented. A drawback of this solution is the ongoing cost that subscribers are required to pay.

1.1.3 Adult web email with child supervision

Although the popular Hotmail (Microsoft Corporation, 2008) has recently included a means for adults to supervise a child's list of contacts the user interface is still designed for adults and includes potentially inappropriate adverts.

1.2 Mifrenz 0.1

Available for download from http://mifrenz.com, Mifrenz 0.1 is a PC based application that allows parents to control the email addresses that their child can send emails to and receive emails from. Some of the major features and design decisions of the application are given here.

1.2.1 Web based or local application?

The objectives of Mifrenz 0.1 were to provide a low cost email application that would meet the needs of parents, yet still be attractive and easy for children to use. To achieve these objectives, it was decided to not attempt to provide a server based application due to the cost of providing a reliable 24/7 solution on a low budget.

A basic design of Mifrenz is therefore that it does not rely on the provision of a web server, and so by default Mifrenz must be implemented as a desktop application. The application relies on a third party email server for message (email) forwarding and receiving. This arrangement means that the cost of providing a production standard server is not borne by the cost of the Mifrenz software.

The literature, MacManus (2007) asks "Which is better, an offline Web App or an online Desktop App?" and goes on to offer both sides of the argument. The traditional advantage of the desktop application is the superior user interface although the introduction of AJAX enabled web applications has reduced this advantage. Web based applications have the advantage of a much easier user 'installation' experience; there is none; but they have a major disadvantage of not being available when no internet connection exists. MacManus suggests that for applications that depend on internet connectivity, taking everything into consideration, Web applications probably offer a superior solution.

Where does this leave an email application for children, where a rich user interface experience is required, yet it is obviously an internet centred activity? The first version of Mifrenz used the Java Web Start technology that provides an automatic installation (one click on a web link) with a resulting desktop application. Unfortunately, Java Web Start did not suit the situation of an application being installed by one user and used by another as the technology did not permit this. Mifrenz 0.1 was eventually distributed as a Java application wrapped in a standard .exe file.

1.2.2 Sending and retrieving protocols

Mifrenz 0.1 uses the SMTP protocol for sending messages and the POP3 protocol to retrieve email messages from the server.

1.2.3 Contact storage

Contact information (name and email address) is stored in a file on the disk drive of the local PC being used. Even before the issue of using multiple computers was raised, problems were encountered with allowing the parent and child to both have write access to the contact data (as was required as either could be logged on to the computer when contact data needed to be created or updated). The requirement of storage that both child and parent had access to was thus already an issue of concern.

1.2.4 SPAM filtering

A major design of Mifrenz is the filtering of messages from unapproved email addresses. The filtering basically involved the deletion of any messages from non contacts. The messages that were not deleted were then placed in the 'inbox' directory on the local PC. Only the contents of this inbox were displayed to the child.

1.2.5 Jokes

To encourage children to use Mifrenz the user interface included the option for children to create, send and collect jokes. These jokes were stored on the local disk drive.

1.2.6 Development environment

Mifrenz was developed using the Netbeans Integrated Development Environment (IDE) (Netbeans, n.d.) and the Java SE Development Kit (JDK) (Sun Microsystems, 2007) programming language. Predefined classes of the JavaMail API (Sun Microsystems, 2008)) were used to access SMTP (Simple Mail Transfer Protocol) and POP3 email servers for sending and receiving email respectively.

2 Problem Identification and motivation

Identifying the problem was an iterative process. True to the framework proposed by Peffers (2006) it was the evaluation of a previous piece of work (Mifrenz 0.1) that identified the initial problem for this work.

2.1 Evaluation of Mifrenz 0.1

During end user evaluation of Mifrenz 0.1, it became clear that a problem arose when a child wished to access their email from multiple locations – a substantial number

of children live with separated parents. This problem arose because Mifrenz was implemented so that all email messages where downloaded from the email server to the local PC (and removed from the server). In situations where a child shared time between two households, when they were at one of the locations, they would not have access to emails that had been downloaded to the other location.

In addition to accessing their messages from multiple locations, was the problem that the contact information was also stored locally and so would not be available at other locations.

A final problem was that any jokes that had been created or collected were also stored locally and so would not be available from another location.

The problem for this current work was therefore how to provide children access to their email, contacts and jokes from multiple locations while still meeting the constraints imposed by providing an application suitable for children.

3 Objectives of a solution

A solution was required that would allow a child to access their email, contacts and jokes from multiple locations while still restricting the email that they send and receive to email addresses that have been approved by their parent. Ideally the solution should build on Mifrenz 0.1 to take advantage of the work already completed.

The solution should not depend on the provision of a dedicated email server.

The objectives can be divided into three categories

1. Access to received and sent messages from multiple locations
2. Access to persistent data (contacts and jokes) from multiple locations.
3. Refactoring of Mifrenz 0.1.

4 Design and development

This section first describes the various designs that were contemplated and eventually chosen. The development of these designs and integration into the Mifrenz application is then described.

4.1 Design

The design started with the consideration of the available technologies that might be applicable to solving the identified objectives. Each of the three objective categories is considered in turn.

4.1.1 Access to received and sent messages from multiple locations

There are really only three technologies to consider, SMTP, POP3 and IMAP as these are the only protocols that email servers provide.

4.1.1.1 SMTP

This has to be used to send emails but it does not keep a copy (accessible by the user) of the sent message on the server. For an application to show messages that have been sent from multiple locations another method must be used. Displaying of sent messages therefore became another 'access to persistent data' problem and is considered in section 4.2.

4.1.1.2 POP3

The ability to access all previously received email messages is possible using POP3 if the option to leave messages on the server is selected. This feature of POP3 is not designed to allow access from multiple email client installations giving rise to the following issues:

- Managing which emails have already been read is difficult.
- No access to sent emails from the other location is provided.
- No sharing of contacts is provided.

4.1.1.3 IMAP

The Internet Message Access Protocol, IMAP, is an alternative protocol for message storage and retrieval that is designed to leave messages on the email server so that they can be available from multiple physical locations.

Although the IMAP protocol is seen as a superior method of accessing email compared to POP3 (Spanbauer, 2006), email server providers have been slow to offer this service due partly to the high storage overhead required of the server. This is because IMAP is designed for users to keep their email on the server rather than download it to their local computer. Google (2008) has recently started to offer IMAP and as Claburn (2007) notes, "It's a nice feature that Google clearly thinks businesses will be interested in". Yahoo! has yet to offer IMAP support and this lack of support is causing frustration amongst its users, "So, now that gmail supports imap (for free), why should I bother keeping a yahoo mail address...", (Yahoo! Answers, 2008).

When SMTP is used in conjunction with IMAP, sent messages are still not made available to users unless also deliberately stored in a location accessible by the user.

4.1.2 Access to persistent data

Contact, Joke and sent message files were originally stored on the local PC hard drive. IMAP does not provide for access to contacts and definitely not jokes! Therefore another solution was required. In addition, the ability to share the sent messages between separate installations of Mifrenz was also required. This section describes two designs that were developed for the persistent storage of this data that could provide multi location access.

4.1.2.1 Google Docs

Google Docs (2008) is a Beta web application from Google that allows you to create and store various document types on a web accessible personal storage space. What makes this interesting for Mifrenz 0.2, is that Google provides a Java API (Application Program Interface) that allows code based creation and reading of stored files. During the development of Mifrenz 0.2 a proof of concept program was written that stored and retrieved contact information in a spreadsheet format on the Google Docs web storage site. The spreadsheet could be created automatically and so this provided an

alternative solution for storing contact and joke information. The services offered by Google Docs and Google Gmail (Google, 2008) therefore provided a path for implementing Mifrenz 0.2.

Unfortunately, Gmail's terms and conditions do not allow accounts to be used by children under 12 years old. Having said that, some parents may wish to use Gmail as they may take the view that as their child does not actually have access to the Gmail password when using Mifrenz it is the parent who is using Gmail. To avoid any issues with the Gmail terms and conditions, and to avoid relying on a single IMAP service provider, the Google Docs solution was not adopted.

4.1.2.2 IMAP messages

IMAP does not describe either storage or retrieval of contacts. This means that an email server that provides an IMAP service cannot provide a contact service over the IMAP protocol. This has been raised as an issue in a number of discussion forums on the web, showing the common frustration with this protocol from IMAP users. Users of the popular email client called Thunderbird (Mozilla, 2008) can now install an extension (SyncKolab, 2008), that is still in beta testing stage, that utilizes a message folder to store contact information. Each time the user runs Thunderbird, the locally stored contacts are synchronized with a message folder on the IMAP server that actually stores contact information in the body of the message.

The idea of storing contact information in an IMAP message was developed into a design for storing all persistent data for Mifrenz 0.2. The design specified that contacts were represented by messages in an IMAP email server folder called mifrenzcontacts. A number of predefined properties (accessible via methods) can be stored in a Message object such as who the message is being sent to and from. However, as a message is not designed to store contact information some of the contact details had no convenient field to store the required data. The method 'setRecipients' using the constant 'Message.RecipientType.TO' was used to store the email address of the contact (only one address per contact permitted), and could be used as a unique field. The 'setFrom' method was used to store the email address of the child using the software and the 'setSubject' method was used to store the username of the same child. The remaining contact information (first and last name of contact, and the approval status of the contact were stored using the 'setText' method. Approval status refers to whether or not a parent has given approval for a child to use a contact. The setText method is the method that is normally used to store the words that a person types into an email. In this case each of the values were stored on a separate line in order to make extracting the values back out simply a matter of reading each line in turn.

Unlike Thunderbird, the contacts in Mifrenz need to be accessed by both the child and the parent. So keeping in mind the previously mentioned issue of providing local storage access, it was decided to implement a solution that had no local storage of either messages, contacts or jokes. Storage of the Mifrenz user accounts was still local, as they contained the email server configuration data required to access the server e.g. username, password, server address and port numbers.

As described earlier, before new emails are displayed to the child, emails from non contacts are first deleted and only those that are left are stored in a folder that the child sees. This was seen as a robust design, and it was decided to replicate it in Mifrenz 0.2. So, messages in the IMAP inbox are now downloaded to the Mifrenz application, where they are filtered using the contacts list before being saved in a new folder (called mifrenzinbox) on the IMAP server. The JavaMail API provides methods for creating new folders on an IMAP server, allowing the automatic creation of this folder. This is important as it is the intention of Mifrenz to be as simple as possible to install. Mifrenz could then display all messages in the mifrenzinbox folder in a similar way to displaying locally stored messages. The sequence of events for displaying received messages is shown in Figure 1.

Figure 1 This UML sequence diagram shows the sequence of events for displaying email messages sent to the child. When a Child first logs, Mifrenz retrieves a list of the child's contacts and then retrieves and displays the messages that have already been saved in the mifrenz inbox. When a child presses the 'Check Email' button, all the messages in the normal Inbox are downloaded, filtered and then saved in the mifrenz inbox. The display is finally updated to reflect any changes in the mifrenz inbox.

4.1.3 Refactoring

Mifrenz 0.1 used objects of a class called Email (specifically designed for this application) to represent email messages. During the development of Mifrenz 0.2 the code was 'refactored' (reworked) to replace all references to the Email class with the JavaMail API Message class. One reason for doing this was because objects of the Message class would now be stored on the server; the JavaMail API obviously does not recognise the Email class. This required further refactoring to be

done as the Email class was being used to store an object of the Contact class, giving easy access to the contact that sent (or was being sent to) an email. It was hoped that using the standard Message class would provide other benefits that come with standardisation and using predefined classes with a range of predefined methods.

4.2 Development

Mifrenz consists of approximately 50 classes grouped into one of 3 packages:

- Presentation package

Deals with the graphical user interface.

- Business Logic package

Applies the rules of the application and provides all access to the data access package classes.

- Data Access package.

Deals with access to data either stored on the local hard disk or the email server.

The three layer structure of the application limits the effects of changes to one class on another class. So when the class that accessed the POP server was replaced with a class that accessed an IMAP server, the rest of the software did not have to be changed. Having said that, as already mentioned, some refactoring was also done. This was mainly the replacement of the Email class with the Message class. Whereas the Email class was written specifically for this application, the Message class is a generic class found in the JavaMail API. Because the Email class had been used to pass data from the Data Access layer through the Business Logic layer up to the Presentation layer, changes were required in a large amount of the existing code. In Mifrenz 0.1, data (such as emails and jokes) was stored on the local disk and the user name of the current user was used to determine its location. In Mifrenz 0.2, a new class called ObjectRepository was created to store the current user's data. When Mifrenz launches, an instance of this class was populated with all the email messages stored on the IMAP server (for this user), along with contacts and jokes. The significance of this ObjectRepository class is that once the data has been downloaded from the email server, the data is stored in the computers RAM, and so readily available to the application for displaying.

5 Demonstration

At the time of writing, the Mifrenz email application for children is in the process of being converted from a POP3 based email application to an IMAP application. Proof of concept of storage of contact information in an IMAP folder has been achieved and integration of this technology is into Mifrenz 0.2 is being completed.

Figure 2 shows a screen shot of the child interface, and Figure 3 shows a screen shot of the parent interface.

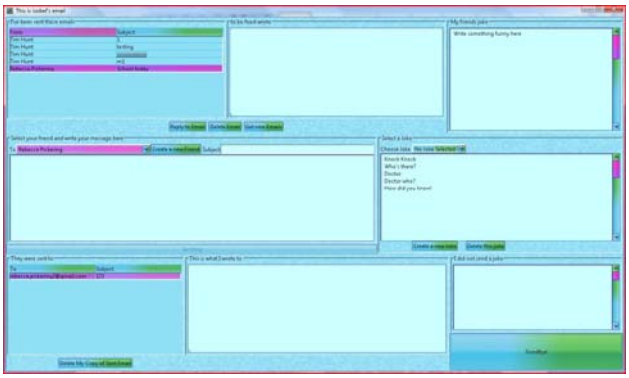

Figure 2: The Mifrenz Child GUI. All the information is available from a single screen making navigation easy.

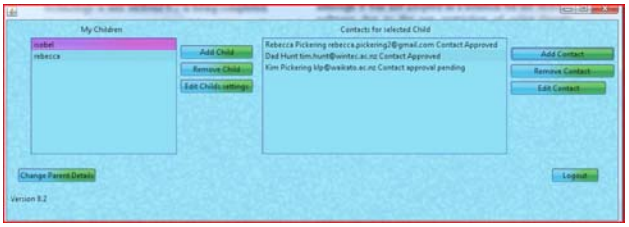

Figure 3: This is the main GUI for the parent. It shows the children that have been added as users, and the contacts for the highlighted child. Child details and contact details can be added or changed by clicking of the appropriate buttons on this screen.

6 Evaluation

Evaluation of Mifrenz will consist of standard software testing practices and will be followed by feedback from end user use.

6.1 Testing

As all the data is stored on the email server, there was a significant delay from when the application was launched to when the users' messages were displayed. Investigations need to be made to see if progressive loading of the data can be achieved (possibly downloading only the message headers first) so giving a more satisfying user experience compared with the current wait while data is downloaded in the background.

6.1.1 IMAP servers

As already mentioned, although Google Gmail provides an IMAP service, Google's terms and conditions restrict usage to children 12 years or older. Another free IMAP email provider that has been tested with Mifrenz 0.2 is @inMail24.com (2008). Also Clearnet (2008) is one of the few New Zealand Internet Service Providers (ISPs) that currently offer an IMAP service.

6.2 Conclusion

IMAP is often seen as a superior (over POP3) method of accessing email when users wish to access their email from multiple computers. Testing of Mifrenz 0.1 showed that children are often in the situation of requiring multi-computer access to their email. It was discovered that although IMAP had advantages, it also had the major disadvantage of not providing a built-in means of storing contact information. This work successfully

demonstrated that contact information could instead be stored in normal message folders on the IMAP server. The use of Google Docs was also demonstrated to be an alternative solution for storing contact information, although it was not adopted as a solution for the Mifrenz software due to the age restriction of using Google Gmail/Docs being incompatible with a children's' email application.

6.3 Future work

A feature of Mifrenz 0.1 is the built in facility for creating and sharing jokes between friends. At the time of writing, this feature had not yet been implemented in the IMAP version. In Mifrenz 0.1 jokes were stored as a binary file (a serialised object) and sent as attachments to emails. Two issues exist with this: 1) the joke cannot be read by non Mifrenz users e.g. a grandparent, and 2) if the Mifrenz Joke class changes, jokes saved as an older version will not be recognised by the new version. Work is underway to solve these issues, and it is envisaged that jokes will be stored as plain text attachments or in the main body of the email message.

7 Communication

The final stage of the Design Science Research Process framework as proposed by Peffers (2006) is the communication of the research – this paper. In addition the new features described in this paper will be incorporated into Mifrenz 0.2 and will be available for download from http://mifrenz.com.

8 Acknowledgements

I would like to thank my colleagues at Wintec who provided invaluable feedback on development of this paper: in particular – Garry Roberton, Dileep Rajendran, Finlay Scott, Hami Te Momo and Susan Bennett. Any errors remaining are wholly the author's responsibility.

9 References

@inMail24.com. (2008). Your own e-mail address and photo album – for free! Retrieved May 9, 2008 from http://www.inmail24.com/Login.aspx.

Claburn, T. (2007). Google Takes No Prisoners. *InformationWeek,(1160), 22.* Retrieved March 8, 2008, from ProQuest Computing database. (Document ID: 1379818961).

Clearnet. (2008), Clearnet beta. Retrieved May 9, 2008 from http://clear.net.nz/

Google. (2008). GMail Beta, Welcome to Gmail. Retrieved April 21, 2008, from http://Gmail.com

Google Docs. (2008). Welcome to Google Docs. Retrieved May 9, 2008, from https://www.google.com/accounts/ServiceLogin?service=writely&hl=en&passive=true&continue=http%3A%2F%2Fdocs.google.com%2F&ltmpl=WR_tmp_2_lfty&nui=1&utm_campaign=en&utm_source=en-et-more&utm_medium=more

Hunt, T.D. (2007). Mifrenz: A new email client application for children. In S. Mann & N. Bridgeman (Eds.) *Proceedings of 20th Annual Conference of the National Advisory Committee on Computing Qualifications* (pp. 99-105). Nelson, New Zealand: NACCQ.

Kidmail.net. (2007) Kidmail.net, The Safe E-Mail Service. Retrieved February 28, 2007, from http://www.kidmail.net/default.asp

Microsoft Corporation. (2008). *Windows Live.* Retrieved April 21, 2008, from http://www.windowslive.com/overview.html

MacManus, R. (2007). Point/Counterpoint: Which is better, an offline Web App or an online Desktop App? Retrieved March 8, 2008, from http://www.readwriteweb.com/archives/offline_webapps_online_desktop_counterpoint.php.

Mozilla. (2008). Thunderbird2. Retrieved May 9, 2008, from http://www.mozilla.com/en-US/thunderbird/

Netbeans. (n.d.). Netbeans IDE 5.5. Retrieved February 28, 2007, from http://www.netbeans.org/

Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V. and Bragge, J. (2006). The Design Science Research Process: A Model for Producing and Presenting Information Systems Research. Retrieved April 6, 2008, from http://ncl.cgu.edu/designconference/DESRIST%202006%20Proceedings/4A_2.pdf

Spanbauer, S. (2006). Drop POP E-Mail for the Freedom of IMAP Servers. *PC World, 24(12), 176*. Retrieved March 8, 2008, from ProQuest Computing database. (Document ID: 1167300961).

Sun Microsystems. (2007). Sun Developer Network (SDN), Java.sun.com, The source for Java developers. Retrieved February 28, 2007, from http://java.sun.com/

Sun Microsystems. (2008). Sun Developer Network (SDN), J2EE JavaMail. Retrieved May 9, 2008, from http://java.sun.com/products/javamail/index.jsp

SyncKolab. (2008). Sync Kolab. Retrieved May 9, 2008, from http://www.gargan.org/extensions/synckolab.html

The IMAP Connection. (2008). Retrieved May 9, 2008, from http://www.imap.org/

Yahoo! Answers. (2008). Retrieved March 8, 2008, from http://answers.yahoo.com/question/index?qid=20071029185456AAWGQAK